Context-based decision-making for virtual soccer players.

Romain Bénard, Pierre de Loor

Context-Based Decision-Making for Virtual Soccer Players

Romain Bénard and Pierre De Loor

CERV European Center for Virtual Reality
25, rue Claude Chappe
BP 38 F-29280 Plouzané (France)
email :{benard, deloor}@enib.fr

Abstract. This article introduces a decision-making model for virtual agents evolving in dynamic and collaborative situations. In order to enhance behaviour credibility and its description, the agent decision-making model is based on notions close to human ones. These notions are context and case based reasoning. After an introduction to dynamic and collaborative situations, we present a formal definition of context adapted to our framework. The next part describes the decision making process. This one relies on the case identification thanks to a graph search algorithm. The last part of this paper illustrates our purpose in the team sport framework, with a result issued from our simulator.

1 Introduction

Our works focus on collaboration learning in dynamic situations [1]. To do so, we develop a virtual environment for training (VET). Humans are immersed in virtual world with autonomous agents, this way they are confronted to situations reflecting reality. Humans are represented by avatars in the simulation. Thus, humans and virtual agents have to collaborate to achieve a task. We introduce, in this paper, a way to give virtual agents autonomy, information retrieval is based on context notion. As context is domain dependent, we decide to follow an application example. We introduce an application of our model in soccer domain. This VET, called CoPeFoot (Collective Perceptions in Football), is populated with virtual agents. We introduce a decision-making mechanism for these virtual agents.

We are starting with the postulate that if agents could have the same mechanisms that those identified by psychologists, on one hand, we can obtain a behavioural credibility and on the other hand a way to make decision-making more explicit. Context is one relevant characteristic identified by psychologist in sport decision-making [2]. The possibility offers by manipulating it in computational models should allow to simulate characteristic behaviours with different expertise levels. Another interesting point is the possibility to test cognitive assumptions in simulated environment such as context influence in a collective behaviour.

Papers dedicated to context place in artificial intelligence are numerous[3,4,5], moreover, this concept is very large. We retain the definition given in [6], that defines context as *a collection of significant conditions and surrounding influences that make a situation unique and comprehensible.* This definition is close to the sport psychologists one considering context as a toolkit for an actor, in situation, to take a decision

In the framework of agent autonomy, context is considered as a perceptions filter. Information are not necessarily relevant and neither in the same manner nor at the same time. An expert defines relevant information according to its point of view. Thanks to this expertise, agent has a perception catalog. It has to find perceptions and construct them when it is in situation.

Moreover, in our approach, agent decision-making relies on the case-based reasoning paradigm [7]. It is based on the assumption that a problem can be efficiently solved by reusing knowledge about already solved cases. Association between context and case-based reasoning has been introduced in [4] and called context-based reasoning. This new paradigm has been used in [5] to implement a personal assistant. Case-based reasoning has been used to set up behaviour of autonomous player for the RoboCup [8], our aim is a bit different we do not look after optimisation but we try to make the decision-making more explicit.

The representation we have chosen, allows to put in place a graph search algorithm to find most similar cases, moreover it allows to have the most explicit case base. This point is very important, case description and reasoning have to be explicit because the trainer should be able to define agents behaviour and to easily set up pedagogical tools as described in [9].

Building such simulator implies some important technical constraints. The first one relies on the number of simulated agents. The decision-making process has to be light in order to be duplicated to simulate up to 22 players in the same simulation. A second constraint is due to the real-time aspect. Decision-making of virtual agents is done under time pressure, moreover agents reaction to context movement has to be visually and temporally credible. Consequently, an agent has to be able to determine the most relevant action at any moment of the simulation. We are using an anytime algorithm [10] based on a description of case base as a tree. Moreover, we have to simulate the environment and perceptions for each agent. To do so, we use a tool ARevi [11][1], a C++ library, allowing to put in place 3D simulations based on a multi-agent approach. Figure 1 illustrates a simulation loop of CoPeFoot (Collective Perception in Football). ARevi provides player physical perceptions. Context plays a role of an active filter on these perceptions to give a semantic according to the domain (soccer in our case). This contextual perception is the base of the identification process to retrieve typical cases, these are defined either by a domain expert or by imitating situation already solved in the simulator.

[1] http://sourceforge.net/projects/arevi/

This document is structured as follows, the next part deals with theoretical frame of our work, in the third part we detail our context definition. The fourth part is focused on decision-making process of our agents. Finally we present a result of changing number and type of context in agent decision-making.

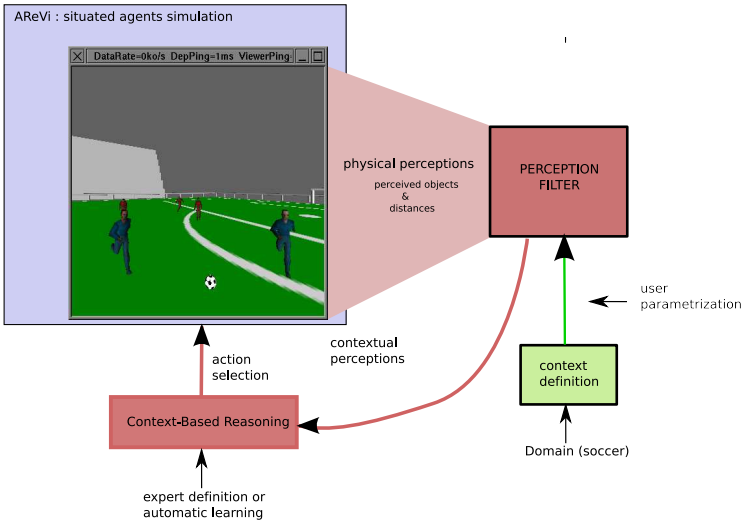

Fig. 1. Context-Based Reasoning in CoPeFoot

2 Collaborative and dynamic situation in sport

Dynamic and collaborative situations can be found in various domains of teamwork with time pressure (rescue, security) or in collective sports [1]. We have decided to keep this last one to illustrate our approach.

More precisely, a dynamic and collaborative situation can be characterized by the following points [12]:

- This implies various protagonists that interact in a common environment and have to solve a problem. The environment state and the protagonists one form the situation. A collaboration between protagonists is needed in order to solve the problem.
- Situation data can be interpreted according to the protagonists point of view. Those agents are able to adopt epistemic point of view on the situation according to their roles.
- Situation interpretation allows a decision making that depends on protagonists objectives. The decision making is materialized by an action or interaction. An action modifies the environment.

– The last point implies that the situation is dynamical. Elements that are kept to take a decision change during the resolution. This evolution is function of the protagonists behaviour, but it is almost linked to environment. This one changes quickly, so the decision has to be taken under time pressure. It is not possible to have complex negotiation mechanisms. This does not exclude all communications type. But, this one is simply brief and often non verbal.

The figure 2 shows a simulation example of a simple collaborative and dynamic situation, the first figure represents the situation: Players *'me'* and *a* are in the same team, their aim is to score. Players *b* and *c* are their opponents.Player *'me'* has the ball. The second figure illustrates a possible solution which consists for the player *'me'* in making a pass to *a*, running behind *b* and calling for the ball. Player *a* passes the ball, and *'me'* just has to score. The solution depicted corresponds to a well known collaboration in soccer called *pass and go*. The last figure is a 3D representation taken from our simulator.

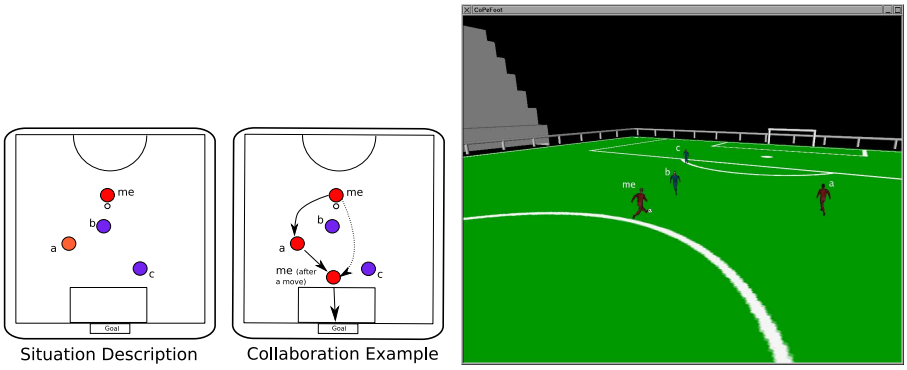

Fig. 2. Example of dynamic situation and its 3D representation in CoPeFoot

3 Context

To give autonomy to virtual agents, one can explore two ways. The first one consists of an exhaustive explanation of behavioural rules of agent, relatively to a global representation of the problem. This is suggested by the use of informed environment [13]. The main problem is exhaustiveness for simulation as complex as human behaviour [14].

The second way consists in modelling agents with internal values with no direct link with the environment. The agent can build its own exteriority by the way of its own world representation [15]. The problem of this approach is the need of interactions abstraction between the virtual environment and its

internal values. It is hard to generate complex autonomous behaviours without direct explicit link with environment.

We argue that context based reasoning can be an alternative between those two ways. It allows a definition of abstractions on perceptions and actions of the agent to let it evolve in an unpredictable environment. It allows to give agent information with a rich semantic as provided by an informed environment. One can take an example of verbalization about an action to define contextual elements. Trainer tells: *If a player has the ball and a partner is closer to the goal than him, he has to pass him the ball unless he is marked* . In the same way a psychologist speaks about the link between the expertise level and the number of context taken into account. We try to formalize this type of knowledge thanks to our contexts. Next part details our context formalization and modeling for our virtual agents.

3.1 Context Structure

An agent context is a set of perceptions. This context stands for the agent own representation of world. Agent decides what action should be executed thanks to its *personal* context. This last is built with all others contexts of the agent as shown on the figure 3. It shows contexts that an agent can have, but not all contexts are its own, some of them can belong to the team or a group in which agent can play a role. Each of these contexts will be explicitly described in the next part. This decomposition allows to enhance our model modularity on the context number. It influences the decision-making and a better adaptation of our model for other implementation than the one introduced here. Thus, psychologists can add or remove a context to estimate its role in collaborative decision-making. Trainer or psychologists can choose the number and type of perception to simulate different expertise levels.

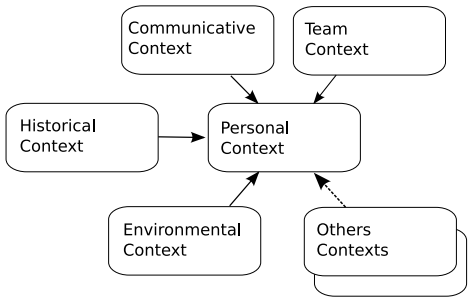

Fig. 3. Personal context building

3.2 Contexts definition and role

In our framework, perceptions have been defined thanks to domain expert help. We are working with sport psychologists, they have done some studies with real soccer players.They have conclued that the basis perceptions of a soccer player are those we present here. In our model, a perception can be seen as predicate which can be satisfied or not. The set of perceptions predicates is called P_{perc} in the rest of this paper. Predicate evaluation is a request to the perception filter which enumerate all conditions that satisfy it. For example, `hasBall(J)` is a perception. This one is true for a player `p` if a player `J` has the ball and `p` has perceived `J` and is aware of this fact. This is due to the fact that a player views only a part of the field, just a part. The request is done in a PROLOG manner, it means the filter answers with all possible values from the simulation at this moment. For this perception, the corresponding value is the player identifier (*i.e* in CoPeFoot: Player.2). If the ball is not perceived by the player, the request will not succeed.

Environmental context is composed of every perception linked to physical environment, for a large part, perceptions linked to the agent field of view. Now, environmental perceptions that are implemented are presented here:

- `distance(+`[2]`CtxObjet3D1, +CtxObjet3D2, ?D)`, true if graphical objects `CtxObjet3D1` and `CtxObjet3D2` are perceived and are at a distance `D`. `D` is a symbolic and qualitative value and $D \in \{nearest, near, far, further\}$
- `relativePosition(+CtxObject3D1, +CtxObject3D2, ?Value)`, true if `Value` represents the position of object `CtxObject3D1` relatively to `CtxObject3D2`. This symbolic value belongs to $\{right, left, front, behind\}$
- `hasBall(?P)`, true if the player `P` has the ball ($\{\exists_{perc}$[3]$(P \in Player, B \in Ball) : distance(P, B, D) \wedge D = nearest\}$).

Communicative context is made up of relevant information coming from messages sent by other agents of the simulation, these can be another player or the referee. Perceptions stored in this context are not messages them self, but their significances. As visual perceptions, their validity is limited in time, time can be adjusted depending on the perception type. Implemented perceptions are:

- `callForBall(?X)`, true if `X` is a partner and he has called for the ball.
- `callForSupport(?X)`, true if `X` is a partner that has asked for support.

Team context does not directly belong to an agent, but to its team. This context is made up of every perception that are shared by every player of the team. Perceptions are:

- `partner(?J)`, true if the perceived player `J` is in the same team (formally $\{\exists_{perc} J \in Player : J.team = agent.team\}$) [4], where *agent* is the perceiving player.

[2] We are using the standard PROLOG notation which allows to indicate if an attribute value is necessarily (+) or if it does not matter (?).

[3] $\exists_{perc}$ means that it exists one perceived element.

[4] Expression `X.Y` refers to the attribute `Y` of object `X`.

- `opponent(?P)`, true if the perceived player `P` is an opponent (formally $\{\exists_{perc} J \in Player : J.team \neq agent.team\}$)
- `isOnAttack(?X)` true if the team of player `X` is on attack. (formally $\{\exists_{perc} J : \{J.team = X.team \wedge HasBall(J)\}\}$)
- `numericalRapport(+X, ?N)` true if `N` is the ratio between the number of players on field in the two teams. For a team `X`, the value belongs to *{weak, equal, strong }*

Historical context allows agent to have a representation of the match history. It corresponds to the previous agent action and some relevant parts of the match evolution. These predicates are:

- `lastAction(+X,?A)`, true if `A` is the last action done by player `X`.
- `timePressure(?T)`, true if `T` is the time before the end of the game, value belongs to *{a lot, enough, few, finishing }*.
- `score(?S)`, true if `S` is a value indicating the actual score in a qualitative manner, value can be *{win, equality, loose}*

As mentionned, the number of our contexts and perceptions are not exhaustive. Thus, the match history could be more refined in order to take into account more previous relevant actions. It can be useful to introduce a group notion during a short period to enhance collaboration. This work should be seen as a base to evaluate context relevance to simulate human behaviour in virtual environment. A second phase of this project consists in increasing the number of contexts and perceptions.

4 Decision-making

Agent decision-making relies on context-based reasoning paradigm [4,5]. Case-based reasoning is often described as a five steps cycle consisting in elaborating, retrieving, reusing, revising and retaining. We have defined in the previous part, how the personal context is built, this corresponds to the elaboration of problem to solve, the first step of the cycle. Revising and retaining is not treated in this paper, but they are one of our goal. This part focuses on the case description and more precisely on the identification mechanism and cases adaptation.

4.1 Cases representation

Cases are stored in a case base. In our approach, base is a tree. Each case is a path from the root to a node. Each node is a perception predicate evaluation. Each node has a table characterizing perception weight for each case in which it appears (figure 4). This weight reflects the importance of the associated perception for this case. Each edge leaving a node contains a possible value of one variable of this predicate (figure 5). This edge allows to go from a perception of a case to the following one.

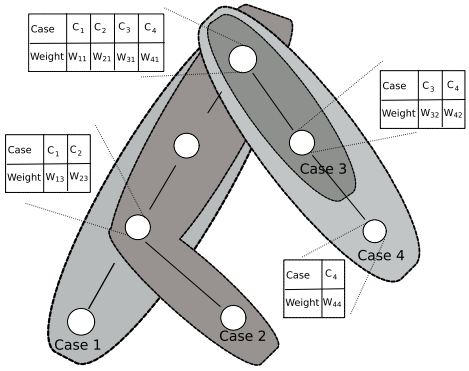

Fig. 4. Each case is a Branch of the tree and has a pertinence weight for each node.

Formally, the tree called $TreeCase$ is a triplet: $TreeCase = \{N, E, C\}$. Where N is nodes set, E edges set and C the cases set. Each case is a branch of the tree associated to an action: $\{\forall c_i \in C : c_i = \{b_i, a_i\}\}$ where $b_i \subseteq N^j$ (where j is the branch depth) and a_i is an action.

Each node is a triplet $\{\forall n_i \in N : n_i = \{pred_i, range_i, pert_i\}\}$ where $pred_i \in P_{perc}$ and $range_i \in \mathbb{N}$. $range_i$ represents the range of the variable to be tested. On figure 5, for the node n_3, $range_3$ corresponds to `D`. Indeed, $pert_i$ is a table representing couples $\{c_k, w_{ki}\}$ where c_k is a case in which actual perception plays a role ($n_i \in c_k$) and $w_{ki} \in \mathbb{R}$ is $pred_i$ weight in the c_k description. Each edge $e_j \in E$ is a test $e_j = \{cond_j, value_j\}$, where $cond_j$ is an operator $cond_j \in \{=, \neq\}$ and $value_j$ is one of the possible values for $range_i$ argument of the predicate $pred_i$.
That is illustrated on the figure 5. Predicate of node n_1 is `hasBall(X)`, edge e_1 is an equality test on the first argument. Constant *'me'* is one of the possible values for argument `X`(representing agent taking decision). In the same way, edge e_4 stands for the third argument of predicate `distance`. *'far'* is one of the possible value for this argument, as noticed in section 3.2. Notice two interesting points : edge e_3 does not correspond to any test and allows just to identify a partner `W`. Second point concerns edges e_1 and e_3 which tests are the same. This can be possible thanks to case representation, a case is a branch or a sub-branch of the tree. Some branches representing different cases can have common perceptions.

This representation has several advantages presented here :

- It can be used as a decision tree allowing an *anytime* identification of perceived situation (case) during simulation as shown in section 4.2.
- It offers a generic abstract representation thanks to variables and predicates utilisation. This mechanism allows to identify an abstract context stored in the base with an agent concrete context. It prevents from a bigger number of

cases in our case base by avoiding symmetry between cases. An unification process allows to affect value to variable as shown in section 4.2.
– It allows to model a generic decision-making. Indeed, action and corresponding case use the same variables. For example, if a previous variable `X` has been unified with player *Player*.4 and the corresponding action is `pass(X)`, agent makes a pass to *Player*.4.

Decision-making simulation credibility depends on tree description. We use two different ways to do so :

1. Expert can specify nodes and edges of the tree, he has to order the tree. To do so, we are currently working with soccer specialists.
2. The second way is based on observation learning. In this case, agent looks how human avatar reacts in simulator and imitates it. Human shows his reaction according to the different situations. A treatment is necessarily to correct mistakes in the tree, this algorithm is based on perception statistic and allows to reorder tree. Some attempt to learn context by observation has already be done, such in [16].
   This learning phase is currently developed and is not detailed in this paper.

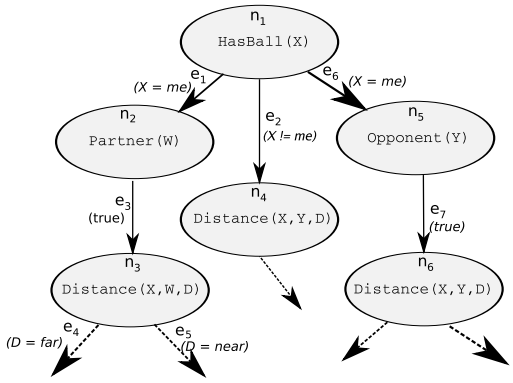

Fig. 5. Case Base representation

4.2 Retrieving a case

Retrieving a case in the base implies to solve the following problems :

– The first step consists of instantiation of tree variable with constants issued from the simulation. Once we have the personal context of the agent, we have to find a perception matching with the perception of the tree root. We say that a perception is matching when we can unify perceptive predicate with

a branch in the graph. If the agent perception and the root one match, we have to continue the tree search. In order to know what is the next perception to test, we are searching correct values of edges leaving the current node to determine next nodes to visit. This instantiation is done with an order defined by the tree. Thus, in the previous example, predicate `hasBall(X)`, according to the simulation state, is unified with the value of the player that has the ball. If the variable is *'me'* (agent who takes the decision), the following node to visit will be n_2. Predicate `partner(W)` is called and try to be unified. Search continues, in the same way, with the node n_5 with the evaluation of the predicate `opponent(Y)`. For the left part of the tree, if a partner has been identified, node n_3 will be activated. Predicate `distance(X,W,D)` has to be evaluated, value of `X` and `Y` are already known, the variable `D` will be unified. A rational search would need a backtracking method to evaluate all possible partners corresponding to `W`. In reality, such exhaustiveness does not correspond to human decision-making. Moreover, it will be technically confronted to research time incompatible with real-time simulation. Selected heuristic consists in unifying variables according to graphical objects relative positions with the agent. This solution is the most pragmatic one, it looks like common sense : nearest objects are the first perceived.

- Next step is the relevance evaluation of the case and selection. Remember that we can have more that one corresponding path.
  To select the best case in the tree, every corresponding case has a global score. It corresponds to the sum of every perception of the case with a reward or a penalty as explained later.

$$score_{c_k} = \{ \sum_{\{i:n_i \in c_k\}} w_k i \} + bonus_k$$

  This score is a compromise between search depth in the tree and the relevance of each perception associated with the case. At the end, for each case every perception weight are summed and the case with the higher score is chosen. Weights allow to take a decision at any time by selecting the case with the highest score. Next algorithm illustrates this purpose. For each evaluated node, the case score is updated by adding the weight of the current perception, this is done by function `updateScore()`.

The stop condition may influence case score.

- Condition 1: All perceptions of agent context have been found in a path, but there is no equality between case and situation context (case has more perceptions than context or there is no time to continue the search). No bonus is given to the score of this case. $bonus_k = \alpha = 0$
- Condition 2: Actual node is a leaf, so we have perfectly identified a case. A bonus is added to the case score. $bonus_k = \alpha$ where $\alpha \geq 0$ is a rewarding parameter, empirically defined.
- Condition 3: One can not find edges leaving the node with the current value. In this case $bonus_k = \beta * n$, where $\beta \leq 0$ is a penalty called correction rate and n is the number of remaining perceptions of the current context.

Algorithm 1: Perceptions tree search algorithm: treeSearch(context)

```
1  begin
2      if (context is empty) then
3          return True // Stop condition 1
4      end
5      nextNodes=[ ] //vector of nodes to visit
6      foreach (perception in context) do
7          if (node predicat and perception can be unified) then
8              nextNodes ⟵ findNextNodes(perception.value)
9              if (nextNodes.size()==0) then
10                 //Stop condition 3
11                 updateScore(score)
12                 return False
13             end
14             else
15                 foreach (node in nextNodes) do
16                     context → delete(perception)
17                     if (treeSearch(context)) then
18                         //recursive call
19                         updateScore(score)
20                         //Stop condition 2
21                     end
22                 end
23             end
24         end
25     end
26     //Every node has been visited
27     return True
28 end
```

Algorithm 2: following nodes search algorithm: findNextNodes(value)

```
1  begin
2      nexts=[ ] //vector of following nodes
3      foreach (edge leaving this node) do
4          if (node.value==value) then
5              nexts→add(node)
6          end
7      end
8      return nexts
9  end
```

An score equality between cases can occur at the end of search, a case between the possible ones is randomly selected. The probability for a case to be selected depends of the number of selection of this case. Only times when the stop condition *2* is reached increases the probability for the case to be selected.

5 Example of results

We introduce in this part a first result from our simulator. An important aspect of CoPeFoot is the possibily to replay a situation, in order to show the important points of the simulated situation, this can be useful for psychologists experience or for training. We track entities position, the following figure shows trajectories of players and ball. We show, on figure 6, a 2D trace from our restitution software.

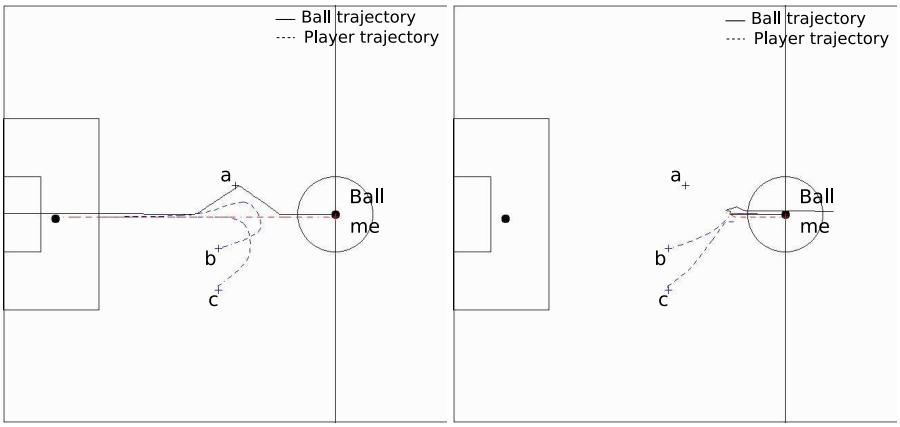

Fig. 6. Examples of agent trajectories

The situation is the one introduced at the beginning of this paper. Players "me" and "a" have to collaborate to score. The first picture illustrates the decision-making and the trajectory of player "me" who has the ball at the beginning. Its decision-making is based on the four contexts introduced earlier. When opponents are too close, it decides to pass the ball to player "a". In the second figure we have deleted the team context, player "me" does not know that it can pass the ball to its partner, because it does not know who is its partner or opponent. The ball is lost and opponents go to the other goal and score. The aim of this example is to show up the possibility of easily degrade agent contexts to experiment influence of different contexts in decision-making.

Thanks to context use, we are able to keep relevant perceptions. We have developped a resititution software in 2D and 3D. The first one allows to keep a trace of entities trajectory. The second one is used to replay a situation. We can show up perceptions as shown on figure 7. This part of software can be either for

Fig. 7. Example of 3D restitution

experimentations or for training, but it can be used to build agent experience. Expert plays a situation and we can show him the result. He can show the perceptions he used to take a decision. According to this result, perceptions relevance can be estimated.

6 Conclusion

We have introduced in this document, that context notion can be seen as a perception filter and as a knowledge representation for virtual reality simulation. The paradigm of case based reasoning can be a good way to model agent behaviour. We argue that context in association with case-based reasoning can provide important elements to set up a virtual environment for training. The first point is the behaviour credibility of agents evolving in our simulator, thanks to simplified analogies with human mechanisms this credibility can be enhanced. The other aspect is to make the agent decision-making process as explicit as possible to explain it to a user. This is possible thanks to context which allows to have a semantic on agent perceptions, the case representation as a path in the perception tree allows to explain choices during the resolution.

Our case base representation, thanks to tree, permits a better visibility for a domain expert. This can ease its integrity verification, and it allows to let an expert set up the base and so agent behaviour. Expert plays in simulator and its action and contexts are stored in the tree. At the end of the demonstration, we can show him the new tree and he can adjust weight of all perceptions for each new case. This work is done in collaboration with sport psychologists that help us to set up some tests in order to validate our approach in the next months.

References

1. M.T. Argilaga and G.K. Jonsson. Detection of real-time patterns in sports: Interactions in soccer. *International Journal of Computer Science in Sport*, Volume 2/Edition 2(1):118–121, 2003.
2. C. Bossard, R. Bénard, and J. Tisseau. Understanding dynamic and collaborative situation: A context based approach. In *Proceedings of the ninth IASTED International Conference Computers and advanced Technology in Education, CATE - 2006, October 4 - 6, 2006, Lima, Peru*, pages 226–231. V. Uskov, 2006.
3. J. McCarthy. Notes on formalizing contexts. In Ruzena Bajcsy, editor, *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 555–560, San Mateo, California, 1993. Morgan Kaufmann.
4. A. J. Gonzalez and R. Ahlers. Context-based representation of intelligent behavior in training simulations. *Trans. Soc. Comput. Simul. Int.*, 15(4):153–166, 1998.
5. A. Kofod-Petersen and M. Mikalsen. Context: Representation and Reasoning – Representing and Reasoning about Context in a Mobile Environment. *Revue d'Intelligence Artificielle*, 19(3):479–498, 2005.
6. J-Ch. Pomerol and P. Brézillon. About some relationships between knowledge and context. In Varol Akman, Paolo Bouquet, Richmond Thomason, and Roger A. Young, editors, *Modeling and Using Context: Third International and Interdisciplinary Conference, Context 2001*, pages 461–464, Berlin, 2001. Springer-Verlag.
7. A. Aamodt and E. Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Commun.*, 7(1):39–59, 1994.
8. A. Karol, B. Nebel, C. Stanton, and M.A. Williams. Case based game play in the robocup four-legged league part i the theoretical model.
9. R. Bénard, P. De Loor, and J. Tisseau. Understanding dynamic situations through context explanation. In *Proceedings of the 6th IEEE International Conference on Advanced Learning Technologies, ICALT 2006, 5-7 July 2006, Kerkrade, The Netherlands*, pages 1044–1046. IEEE Computer Society, 2006.
10. N. Hawes. Anytime planning for agent behaviour. In *Proceedings of the Twelth Workshop of the UK Planning and Scheduling Special Interest Group*, pages 157–166, 2001.
11. F Harrouet, T Jourdan, and E. Cazeaux. *Le traité de la Réalité Virtuelle (3ème éditions). Les presses de l'Ecole des Mines.*, chapter ARéVi. Number 3. September 2006.
12. P. Dillenbourg. What do you mean by collaborative learning? In P. Dillenbourg, editor, *Collaborative-learning: Cognitive and Computational Approaches*, pages 1–19. Oxford: Elsevier, 1999.
13. C. Buche, R.Querrec, P. De Loor, and P. Chevaillier. MASCARET: Pedagogical multi-agents system for virtual environment for training. *Journal of Distance Education Technologies*, 2(4):41–61, 2004.
14. M. E. Pollack and J. F. Horty. There's more to life than making plans. *AI Magazine*, 20(4):71–83, 1998.
15. P-A. Favier and P. De Loor. From decision to action : intentionality, a guide for the specification of intelligent agents' behaviour. *International Journal of Image and Graphics*, 6(1):87–99, 2006.
16. H. Fernlund, A.J. Gonzalez, R. F. DeMara, and M. Georgiopoulos. Learning tactical human behavior through observation of a human actor. *IEEE Transactions on Systems, Man and Cybernetics*, 2005.